A Light Weight Implementation of ECC Cryptosystem on FPGA for nodes in Wireless Sensor Networks

Leelavathi G\textsuperscript{1}, shaila K\textsuperscript{1}, Venugopal K R\textsuperscript{2}

\textsuperscript{1}Vivekananda Institute of Technology, Bengaluru, Karnataka, India
\textsuperscript{2}University College of Engineering, Bengaluru, Karnataka, India

ABSTRACT
The ECC Cryptosystem is implemented on FPGA for sensor nodes that are used in Wireless Sensor Networks in which nodes are FPGA based. Scalar or Point multiplication, the most important operation of ECC is carried out with double and add algorithm which is implemented using Karatsuba Multiplier. The Karatsuba Multiplier designed for basic finite field multiplication operations at lower levels of design increased the overall performance of the cryptosystem in terms of area, speed, operating frequency and power consumption. The processor is optimized for scalar multiplications with Lopez Dahab and Mixed coordinate systems. This work concentrates on lightweight implementation of ECC that is suitable for Wireless Sensor Nodes. The previous implementations mainly concentrates only on Point multiplication whereas in our work we have implemented complete ECC cryptosystem with 40% less device utilization on FPGA Artix-7xc7a100t-2csg324 and the speed achieved is 432MHz.

Keywords: Elliptic curve cryptography, FPGA, Karatsuba Multiplier, Point Multiplication, Wireless Sensor Networks

I. INTRODUCTION
The special needs of Wireless Sensor Networks (WSNs) such as security issues, communication protocols and hardware platforms, require an intense research activity. Different applications employed on the same WSNs environment will have diverse security requirements, inferring the necessity of using dissimilar security algorithms. Encryption is a sensible countermeasure to protect data, although it increases new processing load to the nodes. Conventional microcontrollers found sensor nodes do not provide adequate computation power to process public key cryptographic operations [1][2]. The practice and potentials of FPGAs in sensor node architectures and their applications are explored in [2].

Neal Koblitz and Victor Miller proposed Elliptic Curve Cryptography (ECC) in the year 1985. At present, ECC is the most efficient and preferred PKC system with shorter keys. The security is to concentrate the difficulty of solving Elliptic Curve Discrete Logarithmic Problem (ECDLP)[3]. ECC is attaining popularity since it provides similar security with significantly smaller key lengths. This feature makes it suitable for resource constrained devices like wireless sensor nodes.
Motivation: Now-a-days Field Programmable Gate Arrays are crossing the 28 nanometer CMOS threshold that offer enhanced platforms which adds additional processing competences to a typical sensor node. Due to inadequate resources of the sensor nodes security is considered to be expensive, the speed and area are more notable in WSNs. Thus, in order to increase the speed and minimize area, while utilizing the available resources, it is important to use multiplier which has better performance with respect to security algorithms. Karatsuba multiplier has been employed to allow the encryption and decryption of messages to speed up ECC algorithm with minimum device utilization [2-4].

Organisation: In Section II, research works related to security techniques, public key cryptography and ECC cryptography, different multiplication algorithms are explained. In Section III, implementation is described with Karatsuba multiplier techniques, ECC Cryptography along with Algorithm and Performance Evaluation. Section IV contains Conclusions.

II. EXPERIMENTAL AND COMPUTATIONAL DETAILS

The modules discussed in this section are designed and implemented in VHDL code and tested on ISE 13.2 software using XILINX FPGA Artix-7 xc7a100t-2csg324 device and simulated with Modelsim Simulator. This FPGA is advised for implementation of Public Key Cryptography as it supports different degrees of security and high-speed execution for GF computations. The static power consumption is very less compared to other FPGAs [2].

2.1 Elliptic Curve Cryptography

In the latter decade, the method of hardware implementation of ECC algorithm distinguished a much focused competition, due to essentiality of the constraints i.e., security, speed and area. The performance of an ECC algorithm depends competently on the arithmetic in the fundamental Galois Field (GF). The Galois Fields are prime order fields GF (p) or Characteristic two fields or Binary fields GF (2^m). Both provides equivalent level of security, GF (2^m) is simple and can be implemented in hardware. In the GF, addition and subtraction are equivalently achieved by modulo-2 arithmetic.

An Elliptic Curve over the Binary field GF (2^m) considered is

\[ y^2 + x*y = x^3 + ax + b \]  \hspace{1cm} (1)

Where, \( a, b \in GF (2^m) \), \( b \neq 0 \). Set of points on \( E (GF (2^m)) \) also comprises point O, which is point at infinity.

Point on the elliptic curve is a couple of elements \( x, y \in GF (2^m) \) that fulfill \( P=(x, y) \in E (GF (2^m)) \) of equation (1).

The input data considered in this work is 8 bits. It is encoded to 192 bit point of the elliptic curve \( p(x, y) \). It is projected as \( p(x, y, z) \) before performing encryption. Encryption is carried out with 192 bit point in projective \( p(x, y, z) \) to produce cipher text, which double secures the data. Decoding is performed at decryption end to get back the original text of 8 bits. The point addition and point doubling are performed with different coordinate systems and are discussed in the following section.
2.2 Field Arithmetic and Coordinate Systems

In affine coordinate system a point on the elliptic curve is represented as \( p(x, y) \). To perform field arithmetic operations i.e., point doubling and point addition, field division is required in affine coordinate system. To avoid this time consuming operation, projective coordinate systems are adopted for point doubling and point addition operation which significantly affects the performance of scalar multiplication in turn the whole ECC cryptographic operation [3].

The equation (1) in projective coordinates is given by,

\[
y^2z+xyz=x^3+ax^2z+bz^3
\]

Projective coordinates encompass representing a curve point as triplet \( x,y,z \in E \) (GF \( 2^m \)), i.e., \( p(x, y, z) \). When comparing with different popular projective coordinate systems, the computational cost of Lopez and Dahab (LD) projective coordinate is found to be lesser and most logical for hardware implementations. In this work, Scalar multiplication operations are performed with point doubling in LD and point addition in mixed coordinates.

Point Doubling: It is the addition of a point \( p(x, y) \) on elliptic curve to itself to attain a new point \( q(x, y) \) on the same curve. LD projective coordinates are employed for point doubling in which \( p(x, y) \) is projected to \( p(x_1, y_1, z_1) \). The output is obtained as \( q(x_2, y_2, z_2) \) is performed with following equations:

\[
x_2=(3x_1^2+az_1^4)-8y_1^2x_1
\]

\[
y_2=(3x_1^2+az_1^4)(4x_1^2y_1^2-x_3)-8y_1^4
\]

\[
z_2=2y_1z_1
\]

Point Addition: It is addition of two points’ \( p(x, y) \) and \( q(x, y) \) on an elliptic curve to get another point \( r(x, y) \) on the same elliptic curve. Mixed coordinates is utilized, where one point \( (x) \), is in affine and one more point \( (y) \) in LD projective coordinate. The normal elliptic point \( (x, y) \) is projected to \( p(x_1, y_1, z_1) \) and the other point is in affine i.e., \( p(x_2, y_2) \). Point addition to obtain \( p(x_3, y_3, z_3) \) is performed with following equations:

\[
x_3=(y_2z_3^3-y_3^3z_2^3)(x_2z_1^2-x_1^2)^2(x_1+x_3z_1^2)
\]

\[
y_3=(y_2z_3^3-y_3^3z_2^3)-x_3(y_2z_1^2-x_1z_1^2)2y_1(x_2z_1^2-x_1^2)
\]

\[
z_3=(x_2z_1^2-x_1z_1)
\]

The multiplications involved in Point doubling and Point addition are performed using Karatsuba multiplier which comprises three types of computation divide, conquer and combine [4]. Number of finite field operations required for Group operations are given in Table 1.

<table>
<thead>
<tr>
<th>Group Operations</th>
<th>Squaring</th>
<th>Addition</th>
<th>Multiplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point Addition</td>
<td>3</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Point Doubling</td>
<td>5</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>
2.3 Karatsuba Multiplier

The effective implementations of finite field operations can be established by performing additions, multiplications and inversion that greatly impacts the performance of ECC Cryptosystem. There are several methods to implement the finite field multiplication, out of which Karatsuba algorithm is termed as a fast multiplication algorithm published in the year 1962. In conventional multiplication method, multiplying two integers of \( n \)-bits takes \( O(n^2) \) bit operations. Numerous algorithms are available to improve \( O(n^2) \) multiplication.

Karatsuba multiplication algorithm uses divide and conquer technique and requires \( O(n^{\log_2 3}) \) bit operations, to multiply two integers of \( n \)-bits. Karatsuba algorithm accomplishes multiplication operation by substituting some multiplications with subtraction and addition operations which are less expensive. Due to recursion overhead Karatsuba algorithm is slower than classical multiplication for small inputs and it is more proficient for large numbers. Karatsuba multiplier is used as the finite field multiplier in point addition and doubling modules [4].

2.4 Scalar Point Multiplication

Scalar point multiplication is the process of computing \( Q = k.P \), where \( Q, P \in E(\text{GF}(2^m)) \) and \( k \) is a scalar value. It is performed by repetitive point additions and doubling which is the basic cryptographic operations. There are numerous algorithms for performing Elliptic Curve Scalar Point Multiplication. Scalar Point Multiplication performed in the design is given in Algorithm 1. The multiplication module used [5] in our design is shown in the Figure 1. Basically, it is double and add method, built on the binary expansion of the integer \( k \). In this algorithm point doubling is performed in each iteration of \( k \). In any iteration, if the particular bit of \( k \) is 1, then a point addition is also performed [4].

Input to the module is point \( P(x, y, z) \) and scalar \( k \), where, \( k \) has a size of \( t \) bits. The output obtained after multiplication is \( K.P \). Multiplication of a scalar value \( K \) with a point \( P \) and \( 2P \) output from point doubling block is used for further cascading of modules. Multiplication module used in our implementation is shown in the Fig.1. The module consists of both cascaded point doubling (PD) and point addition (PA) blocks. A point \( P \) is multiplied in every stage to produce \( 2P, 3P, 4P, \ldots, 2t-1P, 2tP \) [5].

![Figure 1: Scalar Point Multiplication Module[5].](image)

---

20th August 2017, www.conferenceworld.in

Algorithm 1: Scalar Point Multiplication

**Input:** Point on the curve \( P(x, y, z) \),

**Output:** Point on the Curve \( Q(x_0, y_0, z_0) = K \cdot P(x, y, z) \)

\[ Q(x_0, y_0, z_0) = P(x, y, z) \quad // \quad Q = P \]

For \( i = t - 1 \) to 0 do

\[ P(x, y, z) = \text{point double}(x, y, z) \quad // \quad P = 2P \]

If \( K_i = 1 \) then

\[ P(x, y, z) = \text{point add}(x_0, y_0, z_0, x, y, z) \]

End if

End for

Return \( Q(x_0, y_0, z_0) \)

The Scalar Point Multiplication is represented as \( Q = K \cdot P \). It is given by,

\[ K \cdot P(x, y, z) = (2^{t-1} K_{t-1} + 2^{t-2} K_{t-2} + \ldots + 2^2 K_2 + 2 K_1 + K_0) \tag{9} \]

\[ K \cdot P(x, y, z) = (2^{t-1} P(x, y, z) K_{t-1} + 2^{t-2} P(x, y, z) K_{t-2} + \ldots + 2^2 P(x, y, z) K_2 + 2 P(x, y, z) K_1 + K_0) \tag{10} \]

Where, \( P(x, y, z) \) is a Point on the Elliptic Curve represented as projective point in design. \( K \cdot P \) with respect to the Figure 1 is given as:

\[ K \cdot P(x, y, z) = \sum_{i=0}^{t-1} 2^i P(x, y, z) K_i = \sum_{i=0}^{t-1} B_i K_i \tag{11} \]

Where, \( B_i = 2^i P(x, y, z) \) for \( i = 0, 1, 2 \ldots t-1 \).

The Controlled Buffers (CB) generate \( 2^i Pk_i \) for \( i = 0 \) to \( (t-1) \). The bit \( k_i \) of \( K \) and \( 2^i P \) are the inputs to the respective CB. The output of each CB is one of the inputs to the corresponding Point Adder block. Point addition in every stage depends on the value of \( k \) bit. If \( k = 1 \), present value of CB and previous value of CB is added. If \( k = 0 \), then only previous value of the controlled block is added with next stage of CB. The output \( 2^t P \) from the last PD block is shown, where \( 2^t P \) provides cascading with other modules.

### 2.5 Encryption and Decryption

#### 2.5.1 Key Generation

In Elliptic Curve Diffie-Hellman Protocol, the elliptic curve equation and the base point \( P(x, y) \) and are made public. Private key of User A is \( k_A \) which is a scalar value, multiplied to point \( P(x, y) \) to generate public key. User A’s public key is equal to \( k_A \cdot P(x, y) \). Similarly, private key of User B, is \( k_B \) and \( k_B \cdot P(x, y) \) is user B’s public key. The shared secret key (SSK) between two parties A and B is easily calculated by

SSK \((x, y)\)\( = k_A(k_B \cdot P(x, y)) = k_B(k_A \cdot P(x, y)) \tag{12} \]

The Processes involved in Scalar Point Multiplication is represented in Fig 2.
2.5.2 Encryption. The message to be encoded is embedded into the x-coordinate of a point on the elliptic curve \((Pm (xm, ym))\). While user A or user B is sending the message to the other, the shared secret key has to be added to the message \(M(x,y)\) to produce the cipher text \(C(x,y)\).

\[
C(x,y) = M(x,y) + SSK(x,y)
\]  

(13)

2.5.3 Decryption. In decryption the shared secret key is subtracted from the cipher text \(C(x,y)\) to obtain plaintext message \(M(x,y)\).

\[
M(x,y) = C(x,y) - SSK(x,y)
\]  

(14)

III. RESULTS AND DISCUSSION

3.1 Point Addition and Point Doubling

The point addition operation output is shown in Fig.3, displaying a point \(p(x3, y3, z3)\) added to \(p(x4, y4, z4)\) to produce \(p(x5, y5, z5)\).
Point doubling operation output with timing diagram is shown in Fig 4. The point considered for doubling is \( p(x_1, y_1, z_1) \) and the output point obtained is \( p(x_2, y_2, z_2) \).

3.2 Point Multiplication (Scalar Multiplication)

The point multiplication module performs the major operation in ECC. This module includes point addition and doubling operations. Key generation for sender and receiver is performed using the point multiplication. Fig 5 shows synthesized result for Point Multiplication.

3.3 Karatsuba Multiplier

The Karatsuba Multiplier plays an important role in multiplying two 192 bit data in point addition and point doubling operations. The timing diagram with results is shown in Fig 6. The two input numbers are \( X, Y \) and \( R \) is the result.
3.3. Encryption and Decryption

The Encryption module encrypts the plain text $M(x, y)$ to Cipher text $C(x, y)$. The synthesized result is shown in the Fig 7 and Figure 8 shows RTL schematic and simulation result respectively. The original text i.e., 8bit data (00010101) and cipher text $C(x, y)$ 192 bit are shown in the Fig 8.

The decryption module decrypts the cipher text $C(x, y)$ back to plain text of 8 bit value. The synthesized result is shown in Fig 9. From the cipher text $C(x, y)$ the original text of 8 bit (00010101) is recovered.

3.3. Analysis of the ECC Cryptosystem

William et al., [6] designs a pipelined ASIP for ECC using FPGA, XCV2600E-FG1158-8 the hardware (area) consumed by point multiplier and ASIP are compared with our work. In [7] Khaleel et al., presents Folded Modular Multiplier with Virtex-4 as target device. Comparing the time delay obtained by our work with [7] is Point doubling (2.610µsec), Point Addition (4.720 µsec) and Point Multiplication (1.078 msec) shows that the computation is performed faster and there is no combinational path delay.

The hardware required for implementation for Point multiplication in our work is 40% less than previous implementations. Our experimental results of device utilization are compared with [6-13] in Table 2 and Table 3. There is no combinational path delay for Group and Elliptic Curve Operations which increases the speed of Encryption and Decryption Process. Most of the previous works concentrates on the Point Multiplication rather
than complete ECC system. In this work complete ECC Cryptosystem with 40% less device utilization is achieved with a speed of 432MHz.

Table 2: Comparison of Device Utilization for Point Multiplier

<table>
<thead>
<tr>
<th>Author</th>
<th>Field(m)</th>
<th>FFs</th>
<th>LUTs</th>
<th>Slices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chelton[6]</td>
<td>163</td>
<td>7962</td>
<td>26364</td>
<td>16209</td>
</tr>
<tr>
<td>Rebeiro[12]</td>
<td>233</td>
<td>-</td>
<td>37073</td>
<td>19209</td>
</tr>
<tr>
<td>Hossein[14]</td>
<td>163</td>
<td>-</td>
<td>30895</td>
<td>16544</td>
</tr>
<tr>
<td>Our Work</td>
<td>192</td>
<td>504</td>
<td>9788</td>
<td>690</td>
</tr>
</tbody>
</table>

Table 3: Comparison of Device Utilization for ECC

<table>
<thead>
<tr>
<th>Author</th>
<th>Field(m)</th>
<th>Device</th>
<th>LUTs</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xining</td>
<td>128</td>
<td>XC4VLX60</td>
<td>15168</td>
<td>125 MHz</td>
</tr>
<tr>
<td>Kaleel [8]</td>
<td>256</td>
<td>XC4VLX20</td>
<td>16209</td>
<td>143 MHz</td>
</tr>
<tr>
<td>Our Work</td>
<td>192</td>
<td>XC7A100T</td>
<td>9915</td>
<td>432 MHz</td>
</tr>
</tbody>
</table>

IV. CONCLUSIONS

We have performed both Point Multiplication and ECC on Artix7 FPGA that is advised platform for Public Key Cryptography. The experimental results show that the area utilized by both Point Multiplier and ECC is around 40% less than the previous implementations. Hence, this work suitable for resource constrained devices like Wireless Sensor Nodes.

REFERENCES

5. Shylashree, N., Sridhar, V.;: Hardwar Realization of High Speed Elliptic Curve Point Multiplication. In:


